Natural Language Processing with Neural Networks

Qing Ma
Communications Research Laboratory
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
qma@crl.go.jp

Abstract

With learning-based natural language processing (NLP) becoming the main-stream of NLP research, neural networks (NNs), which are powerful parallel distributed learning/processing machines, should attract more attention from both NN and NLP researchers and can play more important roles in many areas of NLP. This paper tries to reveal the true power of NNs for NLP applications as supervised or unsupervised learning devices by concretely introducing three practical applications: part of speech (POS) tagging, error detection in annotated corpora, and self-organization of semantic maps.

1 Introduction

Neural-network-based research in natural language processing (NLP) [2, 4, 10, 18, 20, 21, 30, 31, 35, 40, 41, 42, 45] has a rather long history, extending back to the early 1980s with papers on implementing semantic networks [15], word-sense disambiguation [5], anaphora resolution [36], and syntactic parsing [43]. Compared to this long history, however, the impact of neural networks (NNs) on the whole of NLP research is still very limited, and the usefulness of NN has still not been widely or clearly recognized in the NLP field. One conceivable reason is that the scale of the problems treated in most NN-based NLP research so far has usually been too small to demonstrate the real power of NNs for dealing with practical NLP problems. Or, in some cases, the research has focused mainly on modeling human language processing. Another possible reason may be based on the old prejudice that NNs cannot deal with real-world problems, which are usually very large and very complex, due to the inherent NN defect of nonconvergence.

The purpose of this paper is to try to reveal the true power of NN and sweep away some misunderstandings about its applicability to large-scale, complex, real-word problems by concretely describing three practical NLP applications: part of speech (POS) tagging, error detection in annotated corpora, and self-organization of semantic maps. We hope that readers may discover that a conventional perceptron network that has inherent defect of nonconvergence is only one of various types of NN, and that the min-max module (M^3) neural networks [19] does not suffer from this problem, even though it employs perceptrons as basic modules.

The paper is organized as follows. In Section 2 we describe two kinds of NN-based systems, an NN- and rule-based hybrid tagger [23] and another based on M^3 network [24], for POS tagging. We show that these systems perform much better than hidden Markov model (HMM), better than a pure rule-based approach, and at a level comparable to fashionable machine learning methods, such as maximum entropy (ME) and support vector machine (SVM). Section 3 introduces a unique error-detection method [25] using the M^3 network, which is designed for on-line use. That is, detection can be performed while the M^3 network is learning a POS tagging problem. This method is also cost-effective, because the detection is performed not by scanning each word of the whole corpus but by directly "flying at" areas that seem likely to have errors and checking the words in those areas instead. Section 4 describes an NN-based approach to self-organizing semantic maps for Chinese and Japanese, which are visible representations in which Chinese or Japanese words with similar meanings are placed at the same or neighboring points, so that the distance between the points represents the semantic similarity of the words [26]. The effectiveness and essentialness of the proposed NN-based method for creating semantic maps are clarified by comparisons with a conventional clustering technique and multivariate statistical analysis. Section 5 then gives a brief conclusion.

2 Part of Speech Tagging

2.1 Introduction

Words are often ambiguous in terms of their part of speech (POS). POS tagging, which disambiguates

Proceedings of the Language Engineering Conference (LEC'02)
0-7695-1885-0/02 $17.00 © 2002 IEEE

words in the context of a sentence, is an essential technique used in NLP. This technique can also be widely applied to many areas of information processing including pre-processing for speech synthesis, post-processing for OCR and speech recognition, parsing, etc.

There have been many proposals for automatic POS tagging systems that use various machine learning techniques (e.g., [1, 6, 29, 39]). We have also developed a neuro and rule-based hybrid tagger, which reached a practical level of capability in terms of tagging accuracy, requiring less training data than other methods. To further improve our system's tagging accuracy, we can employ two approaches: one is to increase the amount of training data, and the other is to improve the quality of the corpus used for training. Because the hybrid tagger we proposed uses multilayer perceptrons, it suffers from problem of an nonconvergent when we increase the amount of training data. To overcome this inherent drawback, we adopted the M^3 neural network, which can solve large and complex problems by decomposing them into many smaller and simpler sub-problems. This section describes the two kinds of POS tagging systems we have proposed: a hybrid tagger, and an M^3 network tagger. In addition, for the second approach to further improve our system's tagging accuracy, a POS error-detection technique has been developed and is described in the next section.

2.2 POS tagging problem

Suppose there is a lexicon $V = \{w^1, w^2, \cdots, w^v\}$, in which the POSs that each word can serve as are listed, and that there is a set of POSs, $\Gamma = \{\tau^1, \tau^2, \cdots, \tau^\gamma\}$. The POS tagging problem is thus to find a string of POSs $T = \tau_1\tau_2\cdots\tau_s$ ($\tau_i \in \Gamma$, $i = 1, \cdots, s$) with the following procedure φ when sentence $W = w_1w_2\cdots w_s$ ($w_i \in V$, $i = 1, \cdots, s$) is given:

$$\varphi : W^t \rightarrow \tau_t, \tag{1}$$

where t is the position in the corpus of the word to be tagged, and W^t is a word sequence centered on the target word w_t with (l, r) words to the left and right:

$$W^t = w_{t-l} \quad w_t \quad w_{t+r}, \tag{2}$$

where $t - l \geq s_s$, $t + r \leq s_s + s$, s_s is the position of the first word of the sentence. Tagging can thus be regarded as a classification problem by replacing the POS with class and can therefore be handled by using supervised neural networks trained with an annotated corpus.

2.3 POS tagging with a hybrid tagger

Our hybrid system (Figure 1) consists of a neuro tagger, which is used as an initial-state annotator, and a rule-based corrector, which corrects the outputs of the neuro tagger. When a word sequence W^t [see Eq. (2)] is given, the neuro tagger initially outputs a tagging result $\tau_N(w_t)$ for the target word w_t. The rule-based corrector then fine tunes the output of the neuro tagger and gives the final tagging result $\tau_R(w_t)$.

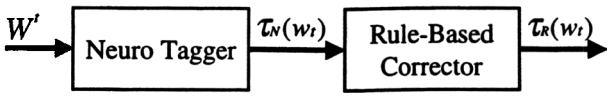


Figure 1: Hybrid neuro and rule-based tagger.

2.3.1 Neuro tagger

The neuro tagger (Figure 2) consists of a three-layer perceptron with elastic inputs.

Input X is constructed from word sequence W^t, which is centered on target word w_t and has length $l + 1 + r$:

$$X = (x_{t-l}, \cdots, x_t, \quad \cdot, x_{t+r}), \tag{3}$$

provided that input length $l + 1 + r$ has elasticity, as described at Sec. 2.5. When word w is given in position p ($p = t - l, \cdots, t + r$), element x_p of input X is a weighted pattern, defined as

$$x_p = g_p \cdot (e_{w1}, e_{w2}, \quad , e_{w\gamma}), \tag{4}$$

where g_p is the information gain which can be obtained by applying information theory (for details see Refs. [27] and [34]), and γ is the number of

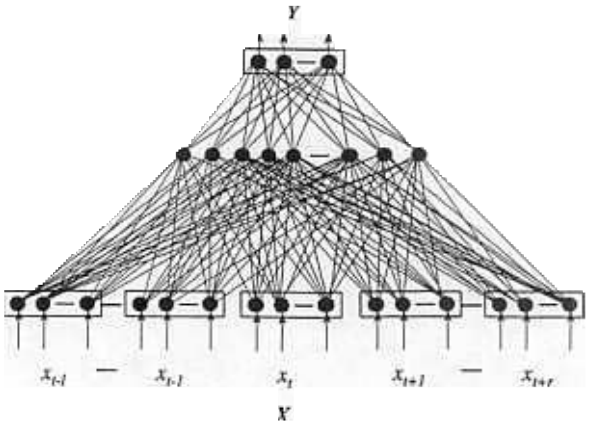


Figure 2: Neuro tagger.

Table 1: Set of templates for transformation rules

Change tag τ^a to tag τ^b when:
(single input)
(input consists of a POS)
1. left (right) word is tagged τ.
2. second left (right) word is tagged τ.
3. third left (right) word is tagged τ.
(input consists of a word)
4. target word is w.
5. left (right) word is w.
6. second left (right) word is w.
(logical AND input of words)
7. target word is w_1 and left (right) word is w_2.
8. left (right) word is w_1 and second left (right) word is w_2.
9. left word is w_1 and right word is w_2.
(logical AND input of POS and words)
10. target word is w_1 and left (right) word is tagged τ.
11. left (right) word is w_1 and left (right) word is tagged τ.
12. target word is w_1, left (right) word is w_2, and left (right) word is tagged τ.

types of POSs. If w is a word that appears in the training data, then each bit e_{wi} can be obtained as follows:

$$e_{wi} = Prob(\tau^i|w), \tag{5}$$

where $Prob(\tau^i|w)$ is a prior probability of τ^i that the word w can take. This probability is estimated from the training data:

$$Prob(\tau^i|w) = \frac{C(\tau^i, w)}{C(w)}, \tag{6}$$

where $C(\tau^i, w)$ is the number of times both τ^i and w appear in the training data, and $C(w)$ is the number of times w appears. If w does not appear in the training data, then each bit e_{wi} is obtained as follows:

$$e_{wi} = \begin{cases} \frac{1}{\gamma_w} & \text{if } \tau^i \text{ is a candidate} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where γ_w is the number of POSs that the word w can serve as. The output Y is defined as

$$Y = (y_1, y_2, \quad , y_\gamma), \tag{8}$$

provided that Y is decoded as

$$\tau_N(w_t) = \begin{cases} \tau^i & \text{if } y_i = 1 \ \& \ y_j = 0 \text{ for } j \neq i \\ Unknown & \text{otherwise,} \end{cases} \tag{9}$$

where $\tau_N(w_t)$ is the tagging result obtained by the neuro tagger.

More information is available for constructing the input for words to the left of the target word, because they have already been tagged. Elastic inputs are used in the neuro tagger so that the length of the context is variable for tagging based on a longest context priority. For details of the perceptron architecture, see Ref. [13], for example. The features of the neuro tagger, i.e., the input elasticity and the utilization of information from left words, are given in detail in Refs. [27] and [28].

2.3.2 The rule-based corrector

Even when the POS of a word can be determined with certainty by only the word on the left, for example, the neuro tagger still tries to tag based on the complete context. In other words, it is difficult for the neuro tagger to learn rules whose conditional parts are constructed by only a single input like $(x_p \rightarrow Y)$. Also, although lexical information is very important in tagging, it is difficult for the neuro tagger to use it, because doing so would make the network enormous. That is, the neuro tagger cannot acquire rules whose conditional parts consist of lexical information. Furthermore, because of convergence and over-training problems, it is both impossible and inadvisable to train neural nets to an accuracy of 100%. The training should instead be stopped at an appropriate level of accuracy. Thus, a neural net may not acquire certain useful rules.

The transformation rule-based corrector makes up for these crucial shortcomings. The rules are acquired from a training corpus using a set of transformation templates by transformation-based error-driven learning [1]. The templates (Table 1) are constructed so as to supply only rules that the neuro tagger has difficulty acquiring, i.e., rules with a sin-

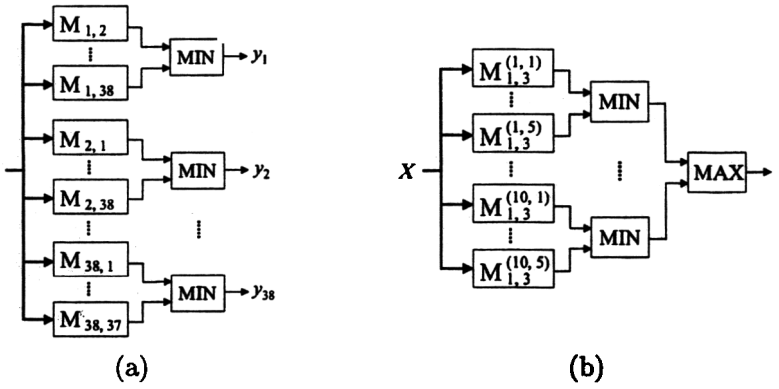


Figure 3: The M^3 Network: (a) the entire construction and (b) a close-up of module $M_{7,26}$

gle input, with lexical information, and with logical AND input of POSs and lexical information.[1] For details on how an ordered list of transformation rules can be acquired, see Ref. [23].

2.4 POS tagging with M^3 networks

This section gives another approach, using an M^3 network, to the POS tagging problem. The central idea underlying the M^3 network is to decompose large and complex problems into smaller and simpler sub-problems, and then combine solutions to the sub-problems to obtain the final solution to the original problem. Since the M^3 network is problem-dependent, in the sense that the problem must be decomposed beforehand, we have to describe how the POS tagging problem is decomposed and how an M^3 network to deal with the problem is then constructed using concrete data, a Thai language corpus, that is actually used in POS tagging experiments.

2.4.1 Decomposition of POS tagging problem

The Thai corpus is composed of 10,452 Thai sentences that have been randomly divided into two sets: one with 8,322 sentences for training and the other with 2,130 sentences for testing. The training set contains 124,331 words, of which 22,311 are ambiguous; the testing set contains 34,544 words, of which 6,717 are ambiguous, in terms of 38 kinds of POSs used in the corpus.[2] For training the M^3 network (as well as the perceptron in the hybrid tagger), only the ambiguous words in the training set were used. By regarding a POS as a class, the POS tagging problem in this case became a 38-class classification problem.

For the M^3 network to learn this 38-class problem, it is first uniquely decomposed into $\binom{K}{2} = 38 \times 37$ two-class problems. Some of the problems (those with over 300 data in this case) are still too large, so these problems are further decomposed by a random method. As a result, the two-class problem $T_{1,3}$, for example, is decomposed into 50 sub-problems, while problem $T_{2,7}$ is not decomposed any further. In this way, the original 38-class problem was decomposed into a total of 3,893 smaller two-class problems. A total of 3,893 modules are thus used in the M^3 network to solve this tagging problem. For the details of decomposing the POS tagging problem, see Ref. [24].

2.4.2 The M^3 network

The M^3 network that learns the POS tagging problem described in the previous section is constructed by integrating modules, as shown in Figure 3(a), where $M_{i,j}$ is used for learning two-class problem $T_{i,j}$. An individual module M_{ij} (e.g., $M_{1,3}$) is further constructed as shown in Figure 3(b) if the corresponding problem T_{ij} (e.g., $T_{1,3j}$, as mentioned in the previous section) is further decomposed.

Input vector X is constructed by Eq. (3) - Eq. (7), except that g_p is set to 1 in Eq. (4) and the output is not fed backed to the input for left words that have already been tagged. Output Y is defined by Eq. (8), provided that it is decoded as

$$\tau_{M3}(w_t) = \begin{cases} \tau^i & y_i \geq 0.5 \text{ and} \\ & y_j < 0.5 \text{ for all j } (j \neq i) \\ \mathit{Unknown} & \text{otherwise,} \end{cases} \quad (10)$$

[1] A number of additional experiments using various sets of templates shown that this set is suitable.

[2] Actually, a total of 47 kinds of POSs are defined in Thai [3]; i.e., $\gamma = 47$.

where $\tau_{M3}(w_t)$ is the POS tagged to word w_t as obtained by the M^3 network.

For the detailed architecture of the M^3 network, see Ref. [19].

2.5 Experimental results

The data: The Thai corpus described in Sec. 2.4.1 was used in our experiments.

Hybrid tagger: The neuro tagger was constructed with a three-layer perceptron whose input-middle-output layers had $L-\frac{L}{2}-\gamma$ units, respectively, where $L=\gamma\times(l+1+r)$ and $\gamma=47$. (l,r) had the following elasticity. In training, (l,r) was increased step by step as (1,1) → (2,1) → (2,2) → (3,2) → (3,3) and gradual training from a small to a large network was performed. In tagging, on the other hand, (l,r) was inversely reduced step by step as (3,3) → (3,2) → (2,2) → (2,1) → (1,1) → (1,0) → (0,0) as needed, provided that the number of units in the middle layer was kept at the maximum value. Learning stopped when the error reached an objective error[3], which was set to 0.005.

By regarding the neuro tagger as already having high accuracy and using the rule-based corrector as a fine tuner, the weight to control the strictness for generating rules in the learning procedure was set to a large value of 100. By applying the templates to the training corpus, which had already been tagged by the neuro tagger, we obtained an ordered list of 520 transformation rules.

The M^3 network: The length (l,r) of a word sequence given to the M^3 network was set at (3,3). All modules were basically constructed by three-layer perceptrons whose input-middle-output layers had $L-2-1$ units, respectively, where $L=\gamma\times(l+1+r)=47\times 7=329$. The modules stopped one round of learning when either the error reached an objective value, set to 0.002, or the number of iterations reached 2,000 epochs. For modules that could not reach the objective error, up to three rounds of relearning was performed, with hidden layers of two units each added in each new round until the objective error was reached.

The results: Table 2 lists the tagging accuracies of various methods. From these results it is apparent that the NN-based methods performed much better than the baseline model[4] and HMM. These methods also performed better than ME and SVM. Among the NN-based approaches, the elastic neuro tagger performed better than the traditional three-layer perceptron in which g_p was set to 1 in Eq. (4) and the output was not fed back to the input for left words that had already been tagged. The M^3 network also performed better than the traditional three-layer perceptron and was comparable to the elastic neuro tagger. We also got the experimental results that the M^3 network hardly experienced any over-training problems, as compared to the conventional perceptron for learning a POS tagging problem. By examining the three factors that determine the learning time, i.e., the number of training data, the number of learning iterations, and the number of parameters to be learned, we can see that the learning time of the M^3 network was more than 60 times shorter than that of the elastic neuro tagger with parallel computation. Without the use of parallel computation, the learning time would have been nearly the same for both. Of all the methods listed in the table, the hybrid tagger had the highest accuracy.

Table 2: Tagging accuracy of various methods'

Method	Accuracy(%)
Baseline	83.6
HMM	89.1
Rule-Based	93.5
ME	92.3**
SVM	93.9**
Three-Layer Perceptron	93.0
Elastic Neuro Tagger	94.4
M^3 Network	94.2
Hybrid Tagger	95.5

*The accuracy was obtained by counting only the ambiguous words.

**The accuracy was obtained without using lexical information.

3 Error Detection in Annotated Corpus

3.1 Introduction

In Sec. 2.1, we mentioned that another approach to improving a POS tagging system is to improve the

[3]Here, the error is defined as $\frac{\sum_{b=1}^{P}\sum_{j=1}^{o}|D_j^{(b)}-O_j^{(b)}(k)|}{oP}$ where $D^{(b)}$ and $O^{(b)}(k)$ are the desired and actual outputs for the b*th* training data at iteration k, respectively, P is the total number of training data, and o is the number of units in the output.

[4]The baseline model performs tagging without using contextual information; instead, it performs tagging with only frequency information: the probability of POS that each word can be.

quality of the manually annotated corpus used for training. For this purpose, we need to use automatic POS error-detection techniques.

POSs annotated manually in corpora may basically have three kinds of errors: simple mistakes (e.g., POS "Verb" is input as "Varb"), incorrect knowledge (e.g., word *fly* is always tagged as "Verb"), and inconsistencies (e.g., word *like* in the sentence "Time flies like an arrow" is correctly tagged as "Preposition", but in the sentence "The one like him is welcome" it is incorrectly tagged as "Verb"). Simple mistake can be detected easily by referring to an electronic dictionary. Mistakes of incorrect knowledge, however, are hardly possible to detect by automatic methods. If we consider tagging words with their correct POSs as a classification or input-output mapping problem of mapping words under the context of POSs, then inconsistencies can be considered as sets of data with the same input but different outputs (classes). These sets of data can then be dealt with by applying existing statistical methods or the neural-network method described in this paper. Previous research on developing a detection technique from statistical approaches [11, 33] was for off-line use; that is, detection had to be performed before learning. Off-line detection, however, is expensive for very large corpora because detection must be performed word by word through the whole corpus, with no preprocessing to first focus on a few areas where the words seem likely to have errors.

This section describes a novel error-detection method [25] based on the M^3 network. This method is designed for on-line use, so that detection can be performed while the M^3 network is learning a POS tagging problem. The method is also cost-effective, because detection is performed not by scanning each word of the whole corpus but by directly "flying at" areas that seem likely to have errors and checking the words in these areas instead.

3.2 New error-detection method

A total of 217 Japanese sentences were selected, each with at least one error, from the Kyoto University Corpus [22] for our computer experiment.[5] These sentences included a total of 6,816 words with 2,410 distinct ones and 97 kinds of POS tags. Regarding a POS as a class, the POS tagging problem in this case became a 97-class classification problem.

For the M^3 network to learn this 97-class problem, it was decomposed into a total of 23,231 smaller two-class problems. The number of training data for each two-class problem was no larger than 80. A total of 23,231 modules were thus used for this learning problem. The input and output vectors were constructed as follows:

$$X = (x_{t-l}, \cdots, x_t, \cdots, x_{t+\tau}).$$

Element x_t is a binary-coded vector with ω dimensions,

$$x_t = (e_{w1}, \cdots, e_{w\omega}), \tag{12}$$

for encoding the target word. Element x_p $(p \neq t)$ for each contextual word is a binary-coded vector with τ dimensions,

$$x_p = (e_{\tau 1}, \quad , e_{\tau\tau}),$$

for encoding the POS with which the word has been tagged.[6] The desired output is a binary-coded vector with τ dimensions,

$$Y = (y_1, y_2, \cdots, y_\tau),$$

for encoding the POS with which the target word should be tagged.

Since an individual module in the M^3 network only needs to learn a very small and simple two-class problem, it can be constructed, for example, with a very simple multilayer perceptron by only using either no or a few hidden units. Thus, an individual module basically does not suffer from nonconvergence, as long as the learning data is correct. In other words, if a module does not converge, it may be concluded that it is learning a data set with some inconsistent data: there is at least one pair of data, (X_i, Y_i) and (X_j, Y_j) in the data set, such that

$$X_i = X_j, Y_i \neq Y_j \quad (i \neq j).$$

Thus, this type of error within an annotated corpus that is being learned may be detected on-line by simply picking out the unconvergent modules and then determining if the data are in conflict with each other. That is, we can use a simple program to find a set of pairs, (X_i, Y_i) and (X_j, Y_j), from the data set the module is learning that satisfy Eq. (15). Picking out the unconvergent modules is clearly just like the process of focusing on areas that seem likely to have errors. Since the number of unconvergent modules will be very limited compared to the number of converged ones when a high-quality annotated corpus is used, and because each module learns a very small

[5] The reason for using the Japanese corpus instead of the Thai corpus is that we have insufficient knowledge of Thai to judge the error-detection results.

[6] This coding has been simplified compared to that used in the POS tagging system described in 2.3.1. Such a simplification, however, does not affect the error-detection mechanism that determines patterns that are in conflict with each other.

Table 3: Experimental results for error detection

Total no. of modules	No. of unconvergent modules	No. of modules with conflicting data	No. of conflicting data	No. of errors
23,231	82	81	97	94

data set, this on-line error-detection method has extremely good cost performance, which will increase with the size of the corpus. By using such an effective on-line error detection method, the quality of the corpus can be improved with a limited manual intervention while it is being learned, and the new data can immediately be used to retrain the unconvergent modules.

3.3 Experimental results

Because there are 30,674 distinct words and 175 kinds of POSs in the whole corpus, the dimensions of the binary-coded vectors for word and POS, ω and τ, were set to 16 and 8, respectively. The length (l, r) of a word sequence was set to (2,2). The number of units in the input layers of all modules was therefore $[(l+r) \times \tau] + [1 \times \omega] = 48$, and all modules were basically constructed from three-layer perceptrons whose input-hidden-output layers had 48-2-1 units, respectively. The modules stopped one round of learning when either the average-squared error reached an objective value of 0.05, or the number of iterations reached 5,000 epochs. For modules that could not reach the objective error, up to five rounds of relearning was performed, with hidden layers of two units each added in each new round until the objective error was reached.

Table 3 lists the experimental results. of the total of 23,231 modules, 82 did not finally converge. Of these 82 modules, 81 had exactly 97 pairs of contradictory learning data, which at first reinforced the hypothesis that the M^3 network has basically no problems of nonconvergence. By checking these 97 pairs of learning data, we found 94 of them included true POS errors, so the precision rate reached nearly 97%. The remaining three pairs, on the other hand, were cases of tagging the Japanese word "*de* (in, at, on, ...)" functioning as either a postpositional particle or copula in various contexts. The word "*de*", however, belongs to a very special case in which it is not sufficient to use only an n-gram word and POS information to determine its POS; instead, the grammar of the whole sentence has to be considered. This result indicates that our method could not only detect POS errors with virtually a 100% precision rate, but also discovered some knowledge that is useful for NLP but would be difficult to find in general.

4 Self-organizing Semantic Map

4.1 Introduction

Computing word similarity in meanings is an important technique that can be applied in many natural language processing fields, such as query expansion in information retrieval [12] and reasoning in word sense disambiguation [7, 16]. A number of corpus-based statistical approaches have been used to compute word similarity [8, 14, 32]. In practical applications, e.g., to find the optimal query expansion in information retrieval, however, words must be further sorted globally based on prior computations of word similarity. For this, we are developing a technique that maps words from a very large lexicon into a small semantic space, i.e., a visible representation called a semantic map in which words with similar meanings are placed at the same or neighboring points so that the distance between the points represents the semantic similarity of the words.

There have been several studies on developing semantic maps for English [37] and also for Chinese and Japanese [26]. In these studies, a self-organizing neural network, called a self-organizing map (SOM) [17], has been adopted as an unsupervised learning machine. In these self-organizing English maps, however, the training data was gathered from only three-word windows and then simply coded (i.e., transformed into vectors) with randomly generated patterns. In studies to develop Chinese and Japanese maps, on the other hand, co-occurrent words, used as training data for the target words, were gathered according to their grammatical relationships, i.e., adjective/noun–noun in Chinese and adjective/nominal adjectival–noun in Japanese. The target words were then coded, taking into account the semantic similarity between words, which was computed by using the co-occurrent words.

This section gives a brief description of the approach used for self-organizing Chinese and Japanese semantic maps (for details see Ref. [26]). The maps created are evaluated numerically in terms of the accuracy, recall, and F-measure, as well as by intuition and comparisons with a clustering method and multivariate statistical analysis.

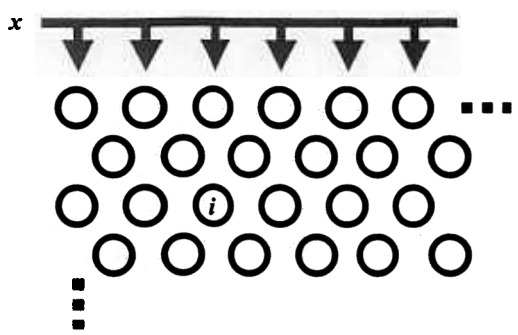


Figure 4: Two-dimensional SOM.

4.2 SOM

An SOM can be visualized as a two-dimensional array (Figure 4) of nodes on which a high-dimensional input vector can be mapped in an orderly manner through a learning process. It is as if some meaningful nonlinear coordinate system for different input features is created over the network. Such a learning process is competitive and unsupervised and is called a self-organizing process. For the details of the SOM architecture, see Ref. [17].

The key point in constructing semantic map based on SOM is how to code the words that we are planning to self-organize into the inputs x shown in Figure 4 based on word similarity computation, which is described in the next section.

4.3 Data coding for self-organizing semantic maps

Suppose there is a set of words w_i $(i = 1, \cdots, n)$ that we are planning to self-organize. Word w_i can be defined by a set of its co-occurrent words as

$$w_i = \{a_1^{(i)}, a_2^{(i)}, \quad , a_{\alpha_i}^{(i)}\}, \tag{16}$$

where $a_j^{(i)}$ is the jth co-occurrent word of w_i and α_i is the number of co-occurrent words of w_i.

Suppose we have a correlative matrix D whose element d_{ij} is the word similarity between words w_i and w_j. We can then code word w_i with the elements in the i-th row of the correlative matrix D as

$$V(w_i) = [d_{i1}, d_{i2}, \cdots \quad d_{in}]^T. \tag{17}$$

$V(w_i) \in \Re^n$ is the input to the SOM. That is, the role of the SOM is to manifest the semantic relationships existing in such high-dimensional vectors and represent them in a two-dimensional space. Therefore, the method of computing the word similarity d_{ij} is a key point for coding.

4.3.1 Previous method

In the previous method, the word similarity d_{ij} between words w_i and w_j is measured by

$$d_{ij} = \begin{cases} \frac{(\alpha_i - c_{ij}) + (\alpha_j - c_{ij})}{\alpha_i + \alpha_j - c_{ij}} & \text{if } i \neq j \\ 0, & \text{otherwise,} \end{cases}$$

where α_i and α_j are the numbers of the co-occurrent words of w_i and w_j, respectively, and c_{ij} is the number of co-occurrent words that w_i and w_j have in common. The word similarity d_{ij} is therefore the normalized distance between w_i and w_j in the context of the number of co-occurrent words that they have in common. The smaller d_{ij} is, the closer w_i and w_j are in meaning.

4.3.2 TFIDF term-weighting method

TFIDF calculation is a well-known term-weighting method [44] that has mainly been used to select important keywords in document classification and information retrieval [33, 38]. The notion of using this calculation to weight the importance of each co-occurrent word is based on the assumption that for a given headword, only the words that frequently co-occur with it but rarely co-occur with other headwords are really important. This approach is also based on the idea that each headword can be regarded as a document, and its co-occurrent words as keywords.

In this method, the word similarity d_{ij} between word w_i and w_j is measured by

$$d_{ij} = \begin{cases} \frac{(T_i - T_{ij}) + (T_j - T_{ij})}{T_i + T_j - T_{ij}} & \text{if } i \neq j \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

where T_i and T_j are the expansions of the numbers, α_i and α_j, of co-occurrent words for w_i and w_j, respectively, and T_{ij} is an expansion of the number, c_{ij}, of co-occurrent words that w_i and w_j have in common. They are calculated by

$$T_i = \sum_{x=1}^{\alpha_i} t_x^{(i)} \quad \text{and} \quad T_{ij} = \sum_{x=1}^{c_{ij}} t_x^{(ij)}, \tag{20}$$

where $t_x^{(i)}$ and $t_x^{(ij)}$ are the TFIDF values of the co-occurrent words $a_x^{(i)}$ $(x = 1, \cdots, \alpha_i)$ of w_i and the co-occurrent words $a_x^{(i)}$ that w_i and w_j $(x = 1, \cdots, c_{ij})$ have in common. These values can be calculated, respectively, as

$$t_x^{(i)} = tf(a_x^{(i)}, w_i) \cdot idf(a_x^{(i)}), \tag{21}$$

and

$$t_x^{(ij)} = tf(a_x^{(i)}, w_i, w_j) \cdot idf(a_x^{(i)}). \tag{22}$$

Here, $tf(a_x^{(i)}, w_i)$ is the co-occurrence frequency of co-occurrent word $a_x^{(i)}$ and word w_i, $tf(a_x^{(i)}, w_i, w_j)$ is the co-occurrence frequency of $a_x^{(i)}$, w_i, and w_j, and $idf(a_x^{(i)})$ is the inverse frequency with which $a_x^{(i)}$ appeares in all headwords:

$$idf(a_x^{(i)}) = \log \frac{n}{df(a_x^{(i)})} + \qquad (23)$$

where, n is the total number of headwords and $df(a_x^{(i)})$ is the number of headwords co-occurring with $a_x^{(i)}$.

The TFIDF value $t_x^{(i)}$ (including $t_x^{(ij)}$) is therefore a weight reflecting the importance of co-occurrent word a_x^i for word w_i. If we consider all co-occurrent words to have the same importance to each headword, then Eq. (19) becomes the same as Eq. (18).

4.4 Experimental results

The data: For Chinese, to evaluate the experimental results more easily and objectively, the headwords (a total of 85 nouns) were selected from six categories in "The Contemporary Chinese Classified Dictionary" [9]. As a new category, we also added several Chinese family names that are not in the dictionary but appear frequently in newspapers. The co-occurent words were adjectives and nouns that, together with the headnouns, formed noun phrases, which were gathered by computer from eleven years of "The People's Daily." The total number of co-occurrent words was 69,030, and there were 22,118 different co-occurrent words.

For Japanese, we used noun phrases composed of adjectives/nominal adjectivals and nouns, which were gathered by computer from eight years of the Mainichi Shinbun newspaper in order of the frequency of co-occurring adjectives/nominal adjectivals. There were 100 nouns, 33,870 co-occurrent words, and 4,023 different co-occurrent words.

The SOM: We used an SOM of a 13×13 two-dimensional array. The number of input dimensions, n, was 85 for Chinese and 100 for Japanese. In the ordering phase, the number of learning steps T was set to 10,000, the initial value of the learning rate $\alpha(0)$ was 0.1, and the initial radius of the neighborhood $\sigma(0)$ was set to 13, a value equal to the diameter of the SOM. In the fine adjustment phase, T was set to 100,000, $\alpha(0)$ was 0.01, and $\sigma(0)$ was set to 7. The initial reference vectors $m_i(0)$ consisted of random values between 0 and 1.0.

The Results: Table 4 shows the results of numerical evaluation for the Chinese semantic map, ranked in order of the F-measure. The results obtained with two other coding methods based on cosine measures are not included, because all the words in the maps using these two coding methods were merged together, so it does not make sense to evaluate them. These results show that the proposed self-organizing method performed better than the clustering technique in its classification ability, and the adaptation of the TFIDF term-weighted coding method was definitely effective.

Table 4: Comparative results of various coding methods and clustering.

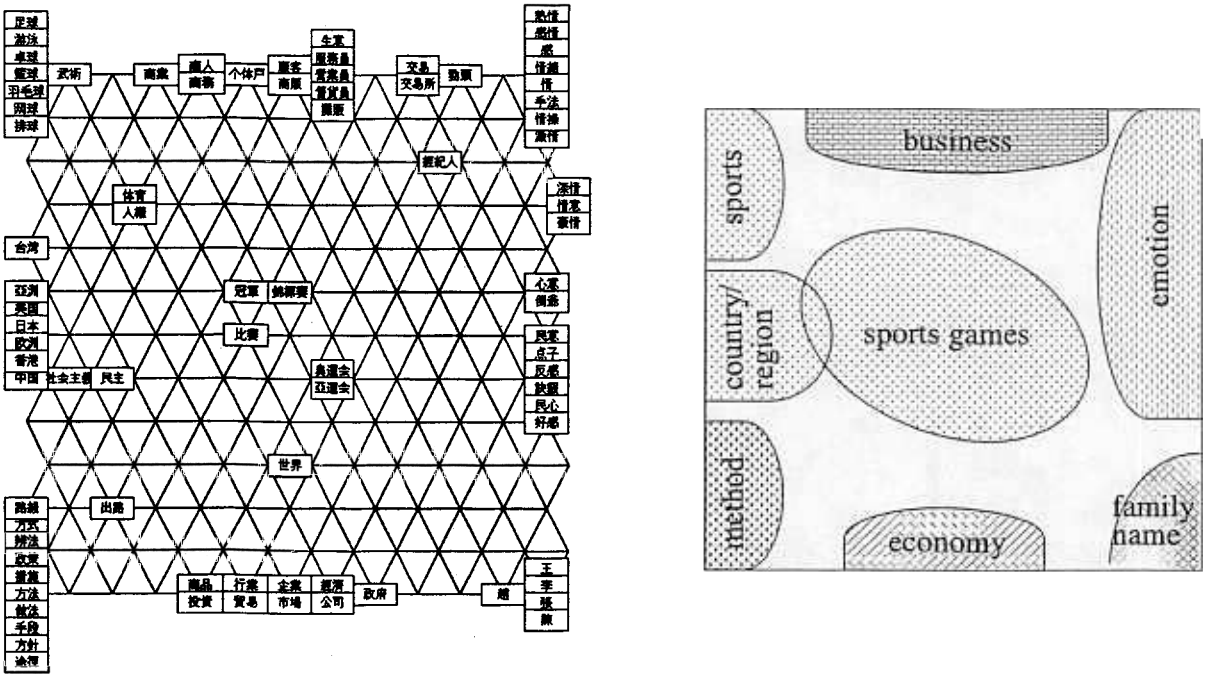
	Precision	Recall	F-measure
Clustering*1	0.936	0.864	0.899
Entropy	0.925	0.874	0.899
Previous	0.926	0.90	0.913
Frequency	0.928	0.90	0.914
Clustering*2	0.95	0.896	0.922
TFIDF	0.944	0.907	0.925

*1Using the previous coding method
*2Using the TFIDF term-weighted coding method

In Figure 5, (a) shows a semantic map of Chinese nouns self-organized by the TFIDF term-weighted coding method, while (b) shows that the nouns in the map can be divided into eight groups according to their similar meanings. Of the 85 nouns, only six nouns were mapped in incorrect areas in the sense that not only were they different from the definition in the dictionary, but also they were also intuitively inconsistent. Even among these nouns, however, some were mapped near the correct area or in locations that are intuitively reasonable. That is, the self-organized map is basically consistent with the definitions found in the Chinese dictionary. Naturally, it is also intuitively consistent, in general.

A comparison of the self-organized map with the classifications obtained by hierarchical clustering shows that both methods produce similar results. Moreover, as with the map, the 85 nouns were also divided into exactly the same eight categories.

Principal component analysis of the same Chinese data by using TFIDF term-weighted coding showed that the cumulative coefficients of determination for the top two and ten principal components were 8.29% and 24.53%, respectively. In general, if the value is less than 80%, the multivariate data cannot be compressed into a small number of principal components. Therefore, it is difficult to construct good semantic maps by applying multivariate statistical analysis. An experiment to plot the data by using the first and second principal components was performed, producing the map shown in Figure 6. Almost all of the nouns were concentrated on the



(a) (b)

Figure 5: Chinese semantic map based on TFIDF term-weighted coding method.

right side, and clearly this method cannot be used to create a meaningful map.

It was not possible to numerically evalute the Japanese maps because all the Japanese nouns were gathered from newspapers by an automatic method and their exact meanings were unknown. We can, however, judge intuitively that the Japanese semantic map (Figure 7) obtained by the TFIDF term-weighting method was better than maps generated with other coding methods, and that the classification was not inferior to that obtained by hierarchical clustering. In addition, principal component analysis of the same data with TFIDF term-weighted coding showed that the cumulative coefficients of determination for the top two and ten principal components were 7.317% and 22.679%, respectively. The plotted results for the first and second principal components showed that all words were merged together, and this method also cannot be used to create a meaningful Japanese map.

5 Conclusions

This paper has demonstrated that neural networks, as powerful supervised or unsupervised learning devices, can be applied effectively to real NLP problems: POS tagging, error detection in annotated corpus, and self-organization of semantic maps. For POS tagging application, we have shown that the NN-based approach performs much better than traditional statistical methods, such as HMM, and it performs slightly better than fashionable machine learning methods, such as ME and SVM. For the error-detection task, we have proposed an on-line approach, so that detection can be performed while the system is learning a problem. This approach is cost-effective, because detection is performed not by scanning each word of the whole corpus but by directly "flying at" areas that seem likely to have errors and checking the words in these areas instead. Lastly, self-organizing semantic maps are an NN-based technique providing advanatges that currently cannot be matched by existing machine learning methods, or by multivariate statistical analysis methods, such as principal component analysis.

For our future work, the proposed tagging method will be expanded for application to word-sense disambiguation, name-entity identification, and dependent analysis, all of which can be regarded as context-based tagging tasks. The proposed error-detection method will be used to check the corresponding corpora used in the various learning tasks decribed above. Finally, the scale of semantic maps will be increased to a level suitable for practical use and then applied to meaning-based information retrieval tasks in Chinese and Japanese.

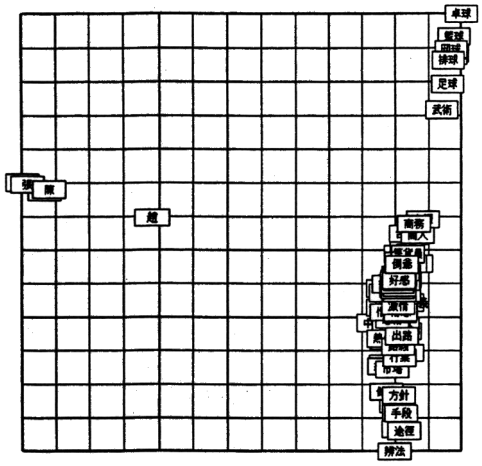


Figure 6: Chinese semantic map using principal component analysis.

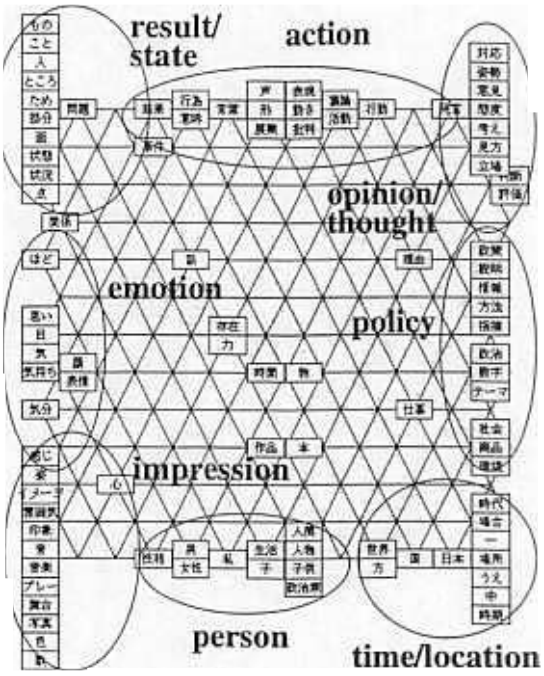


Figure 7: Japanese semantic map based on TFIDF term-weighted coding method.

References

[1] E. Brill, "Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging," *Computational Linguistics*, Vol. 21, No. 4, pp. 543-565, 1994.

[2] N. Chater and M. Christiansen, "Connectionism and natural language processing," in *Language Processing*, S. Garrod and M. Pickering (Eds.), Psychology Press, 1994.

[3] T. Charoenporn, V. Sornlertlamvanich, and H. Isahara, "Building a large Thai text corpus - part of speech tagged corpus: ORCHID," *Proc. NLPRS'97*, pp. 509-512, 1997.

[4] M. Christiansen and N. Chater, "Connectionist natural language processing: the state of the art," Special issue of Cogn Sci on Connectionsit Models of Human Language Processing: Progress and Prospects, Cognitive Science 23, pp. 417-437, 1999.

[5] G. Cottrell and S. Small, "A connectionist scheme for modelling word sense disambiguation," *Cognition and Brain Theory*, 6, 1983.

[6] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis, "MBT: A memory-based part of speech tagger-generator," *Proc. 4th Workshop on Very Large Corpora*, Copenhagen, Denmark, pp. 1-14, 1996.

[7] I. Dagan, S. Marcus, and S. Markovitch, "Contextual word similarrity and estimation from sparse data," *ACL'93*, Columbus, Ohio, pp. 164-171, 1993.

[8] I. Dagan, L. Pereira, L. Lee, "Similarity-based estimation of word cooccurrence probabilities," *ACL'94*, Las Cruces, NM, pp. 272-278, 1994.

[9] D. N. Dong, et al. (Eds.), Contemporary Chinese Classified Dictionary, *Han-Yu-Da-Ci-Dian Press*, 1998.

[10] M. Dyer, "Connectionist natural language processing: a status report," in *Computational Architectures Integrating Neural and Symbolic Processes*, R. Sun and L. Bookman (Eds.), Dordrecht: Kluwer, 1995.

[11] E. Eskin, "Detecting errors within a corpus using anomaly detection," *Proc. NAACL'2000*, Seattle, pp. 148-153, 2000.

[12] William B. Frakes and R. Baeza-Yates (Eds.): Information retrieval: data structures & algorithms, New Jersey, Prentice-Hall, 1992.

[13] S. Haykin, *Neural Networks*, 2nd Edition, Prentice Hall, 1999.

[14] D. Hindle, "Noun classification from predicate argument structures," *ACL'90*, Pittsburgh, PA, pp. 268-275, 1990.

[15] G. Hinton, "Implementing semantic networks in parallel hardwrae," in *Parallel Models of Associative Memory*, G. Hinton and J. Anderson (Eds.), Lawrence Erlbaum, 1981.

[16] Y. Karov and S. Edelman, "Learning similarity-based word sense disambiguation from sparse data," *Proc. the Fourth Workshop on Very Large Corpora*, Copenhagen, pp. 42-55, 1996.

[17] T. Kohonen, Self-organization and associative memory, Springer Series in Information Science, Vol. 8, *Springer*, 1984.

[18] J. Levy, D. Bairaktaris, J. A. Bullinaria, and P. Cairns (Eds.), "Connectionist Models of Memory & Language," UCL Press, 1995.

[19] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification," *IEEE Trans. Neural Networks*, Vol. 10, No. 5, pp.1244-1256, 1999.

[20] H. Isahara and Q. Ma (Eds.), *Proceedings of the First Workshop on Natural Language Processing and Neural Networks (NLPNN'99)*, 1999.

[21] H. Isahara and Q. Ma (Eds.), *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks (NLPNN'2001)*, 2001.

[22] S. Kurohashi and M. Nagao, "Kyoto University text corpus project," *Proc. 3rd Annual Meeting of the Association for Natural Language Processing*, pp. 115-118, 1997 (in Japanese).

[23] Q. Ma, K. Uchimoto, M. Murata, and H. Isahara, "Hybrid neuro and rule-based part of speech taggers," *Proc. COLING'2000*, Saarbrücken, pp. 509-515, 2000.

[24] Q. Ma, B. L. Lu, H. Isahara, and M. Ichikawa, "Part of Speech Tagging with Min-Max Modular Neural Networks," Systems and Computers in Japan, Vol. 33, No. 7, pp. 30-39, 2002.

[25] Q. Ma, B. L. Lu, M. Murata, M. Ichikawa, and H. Isahara, "On-line error detection of annotated corpus using modular neural networks," *Artificial Neural Networks - ICANN2001*, LNCS 2130, G. Dorffner, H. Bischof, K. Hornik (Eds.), Springer, pp. 1185-1192, 2001.

[26] Q. Ma, M. Zhang, M. Murata, M. Zhou, and H. Isahara, "Self-Organizing Chinese and Japanese Semantic Maps," *Proc. COLING'2002*, Taiwan, August, pp. 605-611, 2002.

[27] Q. Ma and H. Isahara, "A multi-neuro tagger using variable lengths of contexts," *Proc. COLING-ACL'98*, Montreal, pp. 802-806, 1998.

[28] Q. Ma, K. Uchimoto, M. Murata, and H. Isahara, "Elastic neural networks for part of speech tagging," *Proc. IJCNN'99*, Washington, DC., pp. 2991-2996, 1999.

[29] B. Merialdo, "Tagging English text with a probabilistic model," *Computational Linguistics*, Vol. 20, No. 2, pp. 155-171, 1994.

[30] R. Miikulainen (Ed.), "Subsymbolic Natural Language Processing," The MIT Press, 1993.

[31] H. Moisl, "NLP Based on Artificial Neural Networks: Introduction," in *Handbook of Natural Language Processing*, R. Dale, H. Moisl, and H. Somers (Eds.), Marcel Dekker Inc., 2000.

[32] S. Mori and M. Nagao, "A stochastic language model using dependency and its improvement by word clustering," *COLING-ACL'98*, Vol. 2, pp. 898-904, 1998.

[33] M. Murata, Q. Ma, K. Uchimoto, H. Ozaku, M. Utiyama, and H. Isahara, "Japanese probabilistic information retrieval using location and category information," *IRAL'2000*, pp. 81-88, Hong Kong, 2000.

[34] J. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

[35] R. Reilly and N. Sharkey (Eds.), "A connectionist Approaches to Natural Language Processing," Hillside, NJ: Lawrence Erlbaum, 1992.

[36] R. Reilly, "A connectionist model of some aspects of anaphor resolution," Proceedings of the Tenth Annual Conference on Computational Linguistics (ACL'84), 1984.

[37] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biological Cybernetics*, 61, pp. 241-254, 1989.

[38] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval," *ACM SIGIR'94*, Dublin, Ireland, 1994.

[39] H. Schmid, "Part-of-speech tagging with neural networks," *Proc. COLING'94*, Kyoto, Japan, pp. 172-176, 1994.

[40] B. Selman, "Connectionist systems for natural language understanding," *Artificial Intelligence Review*, 3, pp. 23-31, 1989.

[41] N. Sharkey (Ed.), "Connectionist Natural Language Processing," Kluwer, 1992.

[42] N. Sharkey, "Connectionist science and natural language: an emerging discipline," *Connectionist Science*, No. 2, 1990.

[43] S. Small, G. Cottrell, L. Shastri, "Towards connectionist parsing," Proceedings of the National Conference on Artificial Intelligence, 1982.

[44] K. Sparck-Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, Vol. 28, No. 1, pp. 11-21, 1972.

[45] S. Wermter, E. Riloff, and G. Scheler (Eds.), "Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing," Springer, 1996.